Fighting Sinkhole Attacks in Tree-based Routing Topologies

Anthonis Papadimitriou, Fabrice Le Fessant, Aline Carneiro Viana, Cigdem Sengul

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Fighting Sinkhole Attacks in Tree-based Routing Topologies

Anthonis Papadimitriou — Fabrice Le Fessant — Aline Carneiro Viana — Cigdem Sengul

N° 6811

January 2009

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--6811--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SACLAY - ÎLE-DE-FRANCE

Fighting Sinkhole Attacks in Tree-based Routing Topologies

Anthonis Papadimitriou , Fabrice Le Fessant , Aline Carneiro Viana , Cigdem Sengul

Thème COM — Systèmes communicants
Équipes-Projets Asap

Rapport de recherche n° 6811 — January 2009 — 21 pages

Abstract:

This work focuses on: (1) understanding the impact of sinkhole attacks on tree-based routing topologies in Wireless Sensor Networks (WSNs), and (2) investigating cryptography-based strategies to limit network degradation caused by these attacks. This work is particularly important as WSN protocols that construct a fixed routing topology may be significantly affected by malicious attacks. Furthermore, considering networks deployed in a difficult to access geographical region, building up resilience against such attacks rather than detection is expected to be more beneficial. We thus first provide a simulation study on the impact of malicious attacks based on a diverse set of parameters, such as the network scale and the position and number of malicious nodes. Based on this study, we propose a single but very representative metric for describing this impact. Second, we present the novel design and evaluation of two *simple and resilient* topology-based reconfiguration protocols that broadcasts cryptographic values. The results of our simulation study show that our reconfiguration protocols are practical and effective in improving resilience against sinkhole attacks, even in the presence of some collusion.

Key-words: wireless sensor network, sinkhole attacks, resilience, tree-based routing protocols

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université
4, rue Jacques Monod, 91893 ORSAY Cedex
Téléphone : +33 1 72 92 59 00

Combattre les attaques par siphon dans les protocoles de routage en arbre

Résumé : Ce travail porte sur (1) l'évaluation de l'impact des attaques par siphon sur les topologies de routage à base d'arbre dans les réseaux de capteurs sans fil (WSNs), et (2) l'utilisation de stratégies à base de cryptographie pour limiter les dégradations du réseau causées par ces attaques. Ce travail est particulièrement important car les protocoles WSN basés sur des topologies statiques peuvent être sévèrement affectés par les nœuds malicieux. De plus, en considérant les réseaux déployés dans les régions difficiles d'accès, les stratégies basées sur la résistance aux attaques peuvent être beaucoup plus efficaces que celles basées sur la détection de ces attaques. Aussi, nous décrivons d'abord une étude par simulations de l'impact d'attaques selon de nombreux paramêtres, tels que la taille du réseau ou la position ou le nombre de nœuds malicieux. Nous appuyant sur cette étude, nous proposons une métrique unique pour représenter cet impact, qui capture de nombreux paramêtres. Ensuite, nous présentons deux protocoles de reconfiguration par diffusion de valeurs cryptographiques. Les simulations montrent que ces protocoles sont pratiques et efficaces pour améliorer la résistance des réseaux contre les attaques en siphons, même en présence de collusion.

Mots-clés : réseau de capteurs,attaque de siphon,routage en arbre

1 Introduction

The deployment of a wireless sensor network (WSN), in general, is governed by its application. One of the most well-known and challenging applications for WSNs is data collection in a difficult to access geographical region, in which the network is expected to operate for a satisfactory period of time without any intervention. In addition, a sensor network is particularly prone to failures and, to be effective, it is necessary to cope with various forms of disruptions, ranging from battery outages to malicious attacks. To cause a disruption, the malicious attacks may not even need to be complicated. By simply propagating false information, malicious nodes can significantly affect network functionality, especially routing. Since malicious attacks are easy to launch, it is essential and, however, extremely challenging, to understand the risk a network is under. This constitutes the first goal of this paper.

For many applications, security (i.e., confidentiality, integrity and availability of information) is vital to the acceptance and use of sensor networks. For instance, a large set of routing protocols in WSNs are based on the construction of a tree-based routing topology initiated by a sink [5, 22, 23, 3, 9, 26, 14]. In particular, these protocols use advertised information (e.g. hop count from a sink) to build a routing topology. To understand how secure operation of these protocols is essential for the health of the network, consider the attack, known as the *sinkhole attack* [8], where malicious sensors pretend to be closer to the sinks than all their neighbors. Attracting more traffic, these sensors can either selectively drop the received data (i.e., *selective-forwarding attack*) or collect sensitive information. Clearly, the protocols that construct a routing topology would be significantly affected by these attacks. More specifically, in Directed Diffusion [5] and TinyOS [14], routes are established simply based on the reception of beacon messages initiated by the sink. Hence, sinkholes are easy to create, even without any collusion among sensor nodes, as there is no mechanism to verify the originator and the contents of the message. Therefore, combating these attacks constitutes the second goal of this paper.

To counter these challenges, this paper first investigates the impact of sinkhole attacks in tree-based routing topologies. We present a simulation study that helps to understand and, consequently, define a good representation of the impact of such attacks. To the best of our knowledge, this is *the first study* on a general metric to represent the impact of sinkhole attacks on the performance of tree-based routing protocols. In particular, we investigate a number of key performance parameters (e.g., distribution, density, positioning, attacker capability) that influence the impact of these attacks. This is a significant challenge due to the high variability of different combinations of parameters. For instance, a low number of malicious nodes that are one hop away from the sink can affect the network in the same way as a high number of randomly distributed malicious nodes. Thus, based on our simulation study, we propose a single metric, which we call "risk factor", that can span these variations.

Finally, in comparison to current work [7, 18, 19, 24], this paper focuses on *resilience* against instead of *detection* of compromised nodes. Resilience is an important property especially in networks deployed in environments where human intervention is difficult. Furthermore, detection mechanisms often introduce more complexity and so, more weaknesses to the system,which do not justify their benefits. Hence, building up resilience may prove to be more beneficial. To this end, as our second contribution, we propose two *RESIlient and Simple Topology-based reconfiguration protocols*: RESIST-1 and RESIST-0. RESIST-1 prevents a malicious node from modifying its

advertised distance to the sink by more than one hop, while RESIST-0 does not allow such lying at the cost of additional complexity.

Via simulations and using our risk factor metric, we studied the performance of RESIST-1 and RESIST-0 (1) for three tree-based routing protocols (2) on a large set of topologies (3) with different levels of vulnerability to malicious nodes. The simulation results showed that our reconfiguration protocols are practical and effective in improving resilience against sinkhole attacks. Finally, we conclude with a discussion on implementation issues, such as the use of cryptography in sensor networks, and on the impact of collusion on our work.

The remainder of this paper is structured as follows. In Section 2 we present the system model. Section 3 investigates the representation of the impact of malicious nodes. In Section 4, we lay out our proposal for two simple and resilient topology-based routing protocols. Performance results are presented in Section 5 and discussions in Section 6. The Section 7 overviews the current literature. Finally, Section 8 concludes with future work.

2 Problem statement

We focus on sensor-based monitoring application scenarios, where accessing the monitored region and/or human intervention is difficult. Therefore, we propose quantifying and limiting the impact of disruptions caused by compromised/malicious nodes. (In the rest of the paper, the terms compromised and malicious are used interchangeably.) In the following, the network and threat models are presented in more detail.

2.1 Network model

We consider a connected WSN consisting of N static sensors randomly scattered on a geographical area, and *only* one sink, each with unique IDs. All sensors are identical in terms of computational, memory, and communication capabilities. In order to keep the focus on the analysis of the functionalities of our proposals, we leave the evaluation of energy consumption for future work. Nodes do not have access to location information. Each node or the sink is able to communicate wirelessly with a subset of nodes (its *neighbors*) that are in its transmission range, r_t. We assume that for any two nodes X and Y, if X can communicate with Y, then Y can communicate with X.

We focus on a large set of routing protocols relying on tree-based topology construction [5, 22, 23, 3, 9, 26, 14]. The data is thus, routed from sensor nodes to the sink through a tree rooted at the sink. The routing tree is an aggregation of the shortest paths from each sensor to the sink based on a cost metric, which can represent any application requirement: hop count, loss, delay among others. Here, the routing tree is built by using the hop distance to the sink.

2.2 Threat model

We focus on sinkhole attacks launched by compromised nodes inside the network. In our threat model, sensors cannot lie about their identities due to the presence of cryptographic measures [20]. We further assume that malicious sensors are not colluding, i.e. collaborating to increase the impact of the attack. The impact of collusion is discussed in Section 6.2.

We assume that public-key cryptographic primitives are available on all sensors (refer to [27] for a survey on key distribution in WSNs). Furthermore, recently it has also been shown that public key infrastructure is viable for WSNs [15]. Hence, in our model, all sensors only know and trust the public key K_{pub}^{sink} of the sink. Additionally, each sensor X has a pair of public-private keys $(K_{pub}^{X}, K_{pri}^{X})$ that it can use to prove its identity. These key pairs can be generated and uploaded offline to the sensors before the deployment. Using these key pairs, nodes perform authentication and sign data messages.

3 Impact of Malicious Sensors

When assessing the performance of tree-based routing protocols, it is crucial to be able to characterize the topology built in terms of its vulnerability to malicious sensors. Typically, "the number of compromised sensors" is used for this purpose [7, 11]. However, this metric is not necessarily indicative of the hazard that malicious nodes might cause in a WSN: one compromised sensor close to the sink can reduce the data delivery success more than dozens of sensors compromised at the border of the network. Intuitively, when tree-based routing protocols are in use, the impact of a malicious sensor mostly depends on the number of uncompromising sensors in its sub-tree. We thus introduce a new metric, called *Risk Factor*, which takes these points into account, and encompasses parameters such as the number of compromised sensors, their position, the density and size of the network. It provides a more accurate way to evaluate the impact of selective forwarding and sinkhole attacks on tree-based routing protocols by enabling the classification of different compromised topologies into a few equivalence classes. Next, we introduce our metric, and show, through different simulations (performed using a discrete event-based simulator implemented in Java), how it captures various parameters of compromised topologies.

3.1 Risk Factor computation

We compute the "Risk Factor" of a given topology by first computing a local risk factor for each node X, denoted as $LRisk_X$. Essentially, $LRisk_X$ intuitively shows the probability that a message from a node X arrives at a compromised sensor on the path to the sink. Then, the risk factor of the whole topology can be computed as the mean of the local risk factors of all nodes in the network.

To compute $LRisk_X$ for all nodes, we first construct the graph $G(V, E)$, where V is the set of sensor nodes and the sink, and E is the set of edges, (i.e. links between nodes that can communicate directly within transmission range). Any shortest path algorithm, e.g. Dijkstra or Bellman-Ford, can be run over $G(V, E)$ to compute the *distance to the sink* for each sensor as the minimum hop count between them. It must be noted that the outcome of the shortest path algorithm is deterministic, unlike minimum spanning trees created by different WSN routing algorithms. This is because the latter depends on the order the nodes are reached during the construction phase, which can greatly vary among separate protocol executions due to different probabilistic factors such as collisions and wireless channel error rate.

The $LRisk_X$ of compromised nodes is selective forwarding dropping probability p, while, $LRisk_{sink}$ is 0, as the sink cannot be compromised. For all other nodes, $LRisk_X$ is computed as the average of the local risk factors of all neighbors that are strictly closer to the sink. More formally:

$$LRisk_X = \begin{cases} 0 & \text{if } X \text{ is the sink} \\ p & \text{if } X \text{ is malicious} \\ \frac{\sum_{Y \in N_X | d_Y < d_X} LRisk_Y}{\|\{Y \in N_X | d_Y < d_X\}\|} & \text{otherwise,} \end{cases} \quad (1)$$

where N_X is the neighbor set of X, d_X is its distance to the sink, $\{Y \in N_X \mid d_Y < d_X\}$ is the subset of its neighbors with a shorter distance to the sink, and $\|S\|$ is the cardinality of S. While Equation 1 does not explicitly represent the attacker capability, except for selective forwarding probability p, the effect of different type of "distance" attacks is captured implicitly through the use of d_X. Note that the "distance attacks", such as the sinkhole attacks considered in this paper, mainly affect how a node perceives its distance to the sink and hence, d_X. We present further detail on risk factor computation under different attacks in Section 5.

$LRisk_X$ is computed recursively starting from the sink and moving to nodes with increasing hop counts. Given $LRisk_X$, $\forall X \in V$, the risk factor of the entire topology, *TRisk*, is:

$$TRisk = \frac{\sum_{X \in V} LRisk_X}{\|V\|} \quad (2)$$

The strength of the proposed risk factor lies in its ability to capture the mean impact of all the possible trees that can be created by an arbitrary routing protocol. Essentially, the local risk factor accounts for all neighbors that are closer to the sink, and hence, it is able to represent all the potential parents (including compromised nodes pretending to be closer to the sink) on any tree-based routing topology.

3.2 Risk Factor pertinence

In this section, we show how our Risk Factor captures the parameters of a compromised topology better than usual metrics. We assume malicious nodes perform selective-forwarding attacks with $p = 1$. The following parameters are studied:

- **Positioning of compromised nodes**, which represents the distribution of compromised nodes in the geographic area covered by the network.
- **Scale of the sensor network**, which defines the number of sensor nodes and the area the network covers.
- **Number of compromised sensors**

3.2.1 Positioning of compromised nodes

The distribution of compromised sensors has an important impact on their hazard. As a rule of thumb, if the compromised nodes are closer to the sink, their effect is more threatening since nodes that are close to the sink carry more data than nodes that are farther. To understand how our Risk Factor takes this into account, we evaluate four different distributions of compromised sensors (not necessarily realistic):

- **Uniformly Random (*UR*)**
- **Linear (*L*)**, so that they form an imaginary line that runs through the area of the network.
- **Ring (*R*)**, so that they form a ring surrounding the sink.

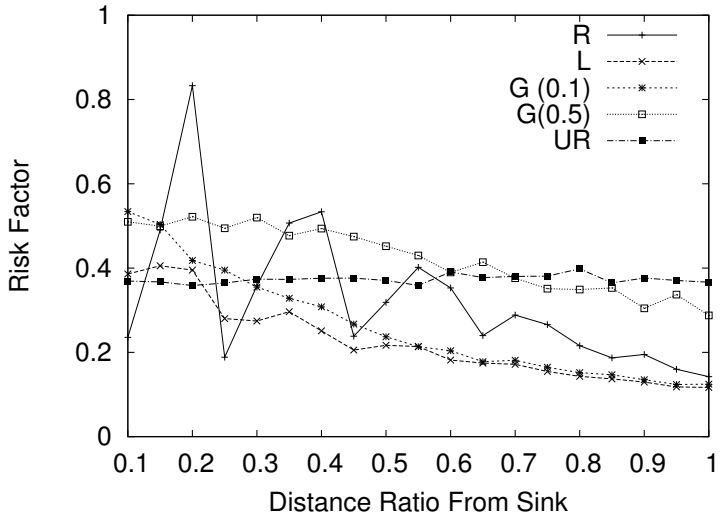


Figure 1: Risk Factor of different positioning scenarios for malicious nodes.

- **Gaussian (*G*)**, so that they follow a Gaussian distribution around a *center point* according to a dispersion parameter.

Fig. 1 shows that our risk factor indeed captures the impact of these distributions. All topologies are of fixed size (500 sensors), density ($3 \ln 500$) (as defined in [4]) and number of compromised nodes (50). All nodes are uniformly distributed in a simulation area $a^2 = \frac{\pi r_t N}{3 \ln N}$ (as in [2]), except the sink which is always in the center. For each distribution, we plotted the risk factor as the *distance from the sink* increases: the definition of this distance is intuitive for R, and does not matter for UR; for L, it represents the distance from the sink to the closest (imaginary) point on the line; finally, for G, it is the distance to the center of the distribution. Note that the distance is normalized by the maximum distance to the side of the network area. For *G*, we use the same normalization for the variance, and restrict ourselves to 0.1 and 0.5 for Fig. 1.

As expected, our risk factor is constant for *UR*, as this distribution does not vary with distance. For the others, the risk factor increases when the distance decreases. Perhaps less expectedly, the risk factor oscillates for the R distribution, which can be explained as follows: if we represent all the nodes with the same distance to the sink as a disk, compromised nodes on the border of the disk have a higher chance than the ones inside of being chosen as parents by the nodes outside the disk. Hence, the risk factor is maximum when the ring of compromised nodes is exactly at a multiple of the transmission range (here, the transmission range is equal to 0.21 times the maximum distance to the sink). Nevertheless, the risk factor for R still globally decreases as the distance increases.

3.2.2 Scale of the Sensor Network

It is important to know how the safety of a network is affected depending on its scale. For instance, networks with higher number of nodes are expected to experience less danger compared to sparse networks for the same number of malicious nodes. Therefore, we define the scale of a sensor network as (1) the number of sensors and (2) the geographical area the sensor network covers. Hence, in our simulations, we evaluated

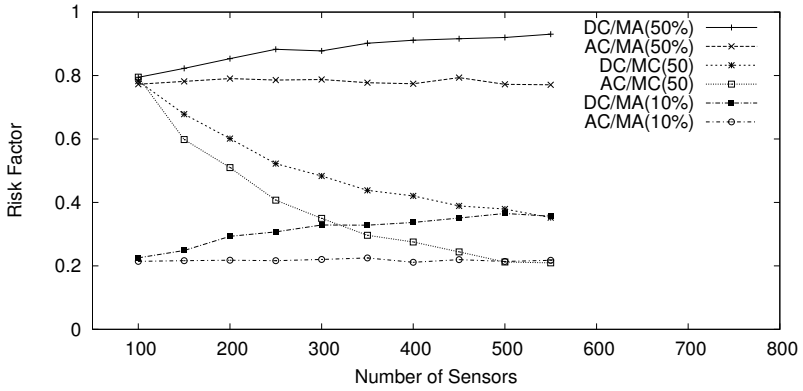


Figure 2: Risk Factor as the scale of the network increases

two different cases when increasing the scale: (1) we kept the area of the network constant (*Area-Constant*/AC deployment) and hence, increased the density by adding more nodes, and (2) increased the area of the network proportionally to the number of sensors (*Density-Constant*/DC). Furthermore, for each case, we first assumed that the number of malicious nodes remained the same (*Malicious-Constant*/MC). Then, we relaxed this assumption by scaling the adversary capability and thus, keeping the uncompromised to compromised ratio constant (*Malicious-Adapting*/MA). In our simulations, in the *DC* deployment, the network density is $3ln(100)$, whereas in the *AC* scenarios the network spans 95×95 meters. In *MC* scenarios, the number of malicious nodes is 50. Finally, we use two *MA* configurations, where the ratio of malicious nodes is 10% and 50%, respectively.

Fig. 2 depicts the risk factor for these different cases. For *Area-Constant* and *Malicious-Constant* (AC/MC), as expected, the risk factor decreases considerably, as the number of nodes increases, since the impact of a constant number of compromised nodes decreases. The same argument also applies to *Density-Constant* and *Malicious-Constant* (DC/MC). However, in the case of *Malicious-Adapting* (MA), the two different deployments exhibit different behaviors. For instance, for *AC*, the increase in the number of nodes is neutralized by the increase in compromised nodes. However, this is not the case for *DC*. Since the transmission range is fixed, a bigger area increases the depth of routing trees to connect nodes to the sink. So, as the number of malicious nodes scales with the number of nodes, each malicious node has a potentially higher impact based on the depth of the tree. The risk factor captures this difference between *AC/MA* and *DC/MA*, as it remains constant for the former and increases for the latter.

3.2.3 Number of compromised nodes

Finally, we present how the risk factor captures the number of malicious nodes in the network. In Fig. 3, we evaluate the risk factor for the different cases of positioning of malicious nodes (discussed in Section 3.2.1). All topologies are networks of 512 sensors with moderate density ($3ln(512)$) and the transmission range r_t of each sensor is 20 m and the network is $185 \times 185\ m$.

Fig. 3 shows that the ring case (i.e., *R*) can cause the major damage to the network with a relatively small number of malicious nodes, however, only at certain distances from the sink (d). On the other hand, for both *UR* and *G*, there is no capacity constraint

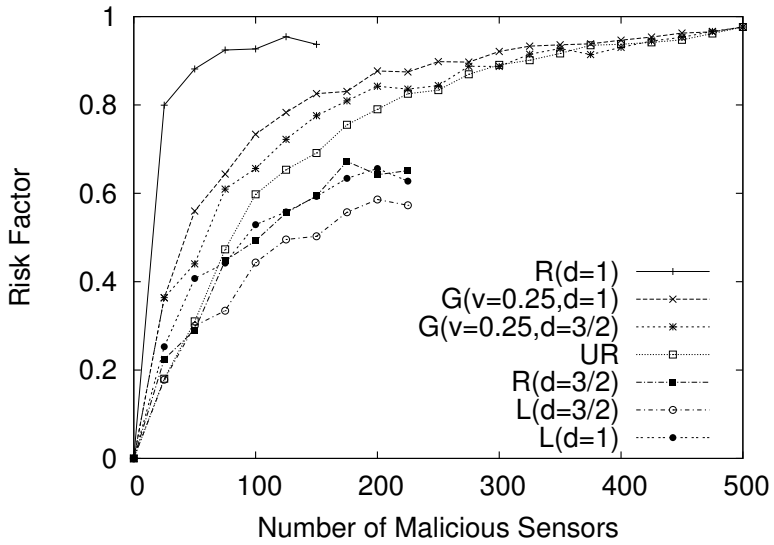


Figure 3: Risk Factor for increasing number of malicious nodes and different positioning : L and R curves are limited by the number of malicious nodes that can be put on the line or the ring.

on the number of malicious nodes. Hence, as the number of malicious nodes increases, their risk factors eventually surpass the risk factor for *R*, where the ring is at a distance r_t from the sink (i.e., $d = 1$). Most importantly, Fig. 3 shows that the risk factor increases fast until $25 - 40\%$ of the nodes are compromised and from this point on, the increase in risk factor is not significant. As this is what would be expected in a real world scenario, we conclude that our risk factor metric is able to represent different topology-related parameters and is able to distinguish their impact successfully.

4 Security protocols

To achieve higher resilience in tree-based routing protocols [5, 22, 23, 3, 9, 26, 14], we propose two schemes to be used at the tree reconfiguration phase triggered by the sink. Note that we do not have special constraints on the period between reconfigurations: it can be chosen either similar to the current tree-based routing protocols or based on cost or topology vulnerability. We define a class of RESIST-h reconfiguration protocols that allow malicious nodes to modify their advertised distance to the sink, but no more than h hops. Based on this definition, we introduce two protocols, RESIST-1 and RESIST-0, which are presented in the remainder of this section. We also describe here cryptographic operations and message contents of the proposed protocols, but refer the reader to Section 6 for an efficient way of implementing them.

4.1 Simple reconfiguration protocol (RESIST-1)

The reconfiguration starts by the sink sending a `Hello`$(epoch, tokens)$ message to all its neighbors, where $epoch$ is a strictly increasing timestamp, chosen by the sink and $tokens$ is a list of tokens $[T_0, T_1, T_2, ..., T_R]$. Essentially, each token is a *(token number,*

epoch) pair signed by the sink:

$$T_k = (k, epoch)_{signed(K_{pri}^{sink})}, \tag{3}$$

where k is the token number. When a sensor receives a `Hello` message, and after verifying that the tokens are correctly signed by the sink (i.e. by using the public key K_{pub}^{sink}), it does the following:

- If the $epoch$ is new, it remembers the identity of the node sending it (his *parent*), and propagates the `Hello` message after removing the smallest token from the list of tokens. In other words, it receives `Hello`$(epoch, [\mathbf{T_k}, T_{k+1}, ..., T_R])$ but sends `Hello`$(epoch, [\mathbf{T_{k+1}}, ..., T_R])$.
- If the $epoch$ is already known, but the `Hello` message advertises a shorter hop distance to the sink (i.e., contains a smaller token), a *selfish approach* would only update the node itself, while a *gossip approach* would also propagate a new `Hello` message to the neighbors. In the rest of the paper, we follow the gossip approach.

Each sensor remembers as its *parent* the sensor that sent it the smallest token signed by the sink for the most recent epoch. Note that the token number of the smallest token is also the hop distance to the sink. Alternatively, sensors can also remember *all the nodes* that advertise the same distance with the smallest token for a given epoch. In Section 5, we also evaluate this approach.

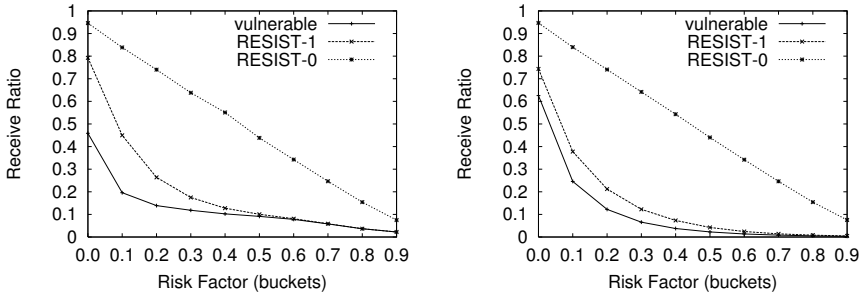
Sinkhole attack resilience: A compromised node can directly forward the `Hello` message without dropping the first token. Assume that the node is the first compromised node on the branch that the `Hello` message travels. Then, if the compromised node is at distance k from the sink, its neighbors would believe they are at distance k too, and so they would believe that the compromised node is at distance $k-1$. Nevertheless, the compromised node cannot pretend to be at a distance smaller than $k-1$, because it would be unable to provide smaller tokens than T_k. Note that as the `Hello` message travels down the tree, it might encounter other malicious nodes that do not drop the token before forwarding the message. In this case, each uncompromised sensor would believe to be at a shorter distance to the sink depending on how many malicious nodes exist before it (e.g., if the number of malicious nodes between the sensor and the sink is 2, then it will *at most* believe it is 2 hops closer to the sink than the reality). Even if we do not have a strong bound on the deviation from the real distance in RESIST-1 (i.e., it increases with the number of malicious nodes on the path), this might not degrade the performance significantly because the main impact is caused by the malicious node closest to the sink.

4.2 Complex reconfiguration protocol (RESIST-0)

This protocol is inspired by a protocol used to measure availability in peer-to-peer networks [10], where newly generated pairs of cryptographic keys are diffused in the network at every round. The sink sends a `Hello`$(epoch, [T_0, T_1, ..., T_R])$ message, where the generated tokens are:

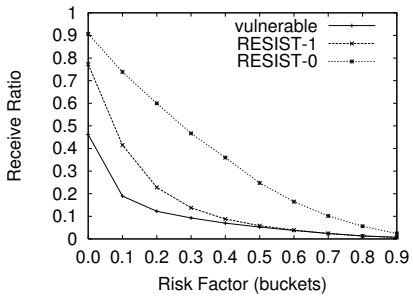
$$T_k = ((k, epoch, K_{pub}^k)_{signed(K_{pri}^{sink})}, (K_{pri}^k)_{signed(K_{pri}^{sink})}), \tag{4}$$

where (K_{pub}^k, K_{pri}^k) is a newly generated pair of cryptographic keys for token k at a reconfiguration $epoch$. The reconfiguration protocol is the same as RESIST-1, except



(a) FTree routing

(b) RRobin routing



(c) RWalk routing

Figure 4: Performance gain for different routing protocols: (a) Fixed Tree, (b) Round-Robin with, (c) Random-Walk ($n = 1$)

that, at the reception of a new epoch and before choosing a sensor Y as its parent, a sensor X challenges the sensor Y first by sending a `Challenge`$(k, epoch)$ message. Basically, this message asks Y to prove its distance k from the sink (i.e. that it has a copy of the token T_k). Sensor Y replies with a message `ChallengeReply`, which contains:

$$((k, epoch, K^k_{pub})_{signed(K^{sink}_{pri})}, (ID^Y, ID^X)_{signed(K^k_{pri})})$$

$(k, epoch, K^k_{pub})_{signed(K^{sink}_{pri})}$ is the first half of the token T_k that Y received. At the reception of the `ChallengeReply` message and using the public key K^{sink}_{pub}, node X can first verify if the token k was correctly signed by the sink. In addition, node X recovers the public key of the token k, K^k_{pub}. Then, it can decrypt the second part of the `ChallengeReply` message, which was encrypted by the private key of token k, K^k_{pri}. Node X can thus verify if its identity, ID^X, was correctly signed by node Y. If so, it believes in ID^Y and in the Y's advertised distance k from the sink.

Sinkhole attack resilience: Since a sensor can sign the second part of the `ChallengeReply` message, if and only if it knows the private key for the token k, it is impossible for a compromised sensor (without collusion) to correctly reply to a `Challenge`. Furthermore, compromised nodes cannot even carry out the attack that we described for RESIST-1. Essentially, not dropping the smallest token would fail, because they would not be able to respond to the `Challenge` for the shorter hop count. Hence, RESIST-0 provides strong resilience against sink-hole attacks. We discuss the impact of collusion on our protocols in Section 6.

5 Performance Evaluation

The goal of our evaluation is to measure the amount of resilience obtained by RESIST protocols described in Section 4. To this end, we present performance results under malicious attacks using three baseline routing protocols, described here after. The experiments were run in a discrete event-based simulator implemented in Java. as we are only interested in a RESIST's algorithmic evaluation, our simulator uses a simplified MAC layer, where neither message losses, nor collisions are considered. In fact, the correct performance of the presented schemes are independent of the order of message arrival.

5.1 Simulation Setup

This section describes the three baseline protocols and our simulation setup. We consider a data collection application, where each sensor periodically sends data (e.g., measurements) to the sink. The routing topology to reach the sink is regularly reconfigured [14]. Malicious nodes do not generate data and they drop every received message with probability $p = 1$.

In our custom simulator, we implemented three baseline routing protocols: *FTree*, *RRobin* and *RWalk*. We studied the performance of these protocols in networks when resilient reconfiguration schemes are used (RESIST-1 and RESIST-0) and not used (*vulnerable* case). In our simulations, compromised nodes try to attract higher volumes of traffic by advertising shorter paths.

In *FTree*, the routing tree is built once at each topology reconfiguration phase. Every sensor forwards all its data to its parent until the next reconfiguration. *RRobin* differs from *FTree* as each sensor computes a set of alternative parents during the topology reconfiguration. This set includes the neighbor that sent the first `Hello` message and any neighbor that sent a `Hello` message with a hop count smaller or equal to the first neighbor. Each time a sensor has to send a message, it selects one parent from this set in a round robin way. In *RWalk* protocol, each sensor makes a random decision about forwarding a message either over the routing tree (computed as in *FTree*) or forwarding it to a randomly selected neighbor. If the message is not sent over the tree, it follows a n-hop random walk and after n hops, it is again forwarded over the tree. The goal of both *RRobin* and *RWalk* protocols is to allow escaping regions that may be severely affected by malicious nodes. In all our experiments with *RWalk*, we use $n = 1$.

We generated many random topologies, progressively filling them with malicious sensors. The space of topologies was divided in 10 *buckets*, where buckets 0, 1, etc. contain the topologies whose risk factor is respectively in [0,0.1), [0.1, 0.2), etc. At each step, the risk factor was evaluated and the topology added to the corresponding bucket, until every bucket had at least 100 topologies. Using these topologies, the performance gain was computed as the ratio of messages that actually reach the sink compared to the number of messages that should reach the sink if no sensor were compromised.

5.2 Resilient Reconfiguration Protocols

In this section, we evaluate each routing protocol separately under different levels of vulnerability to malicious nodes. Our results show that RESIST-0 achieves significant performance gain for all routing protocols (see Fig. 4). RESIST-1 improves performance compared to the vulnerable case, but the gain is much smaller than with

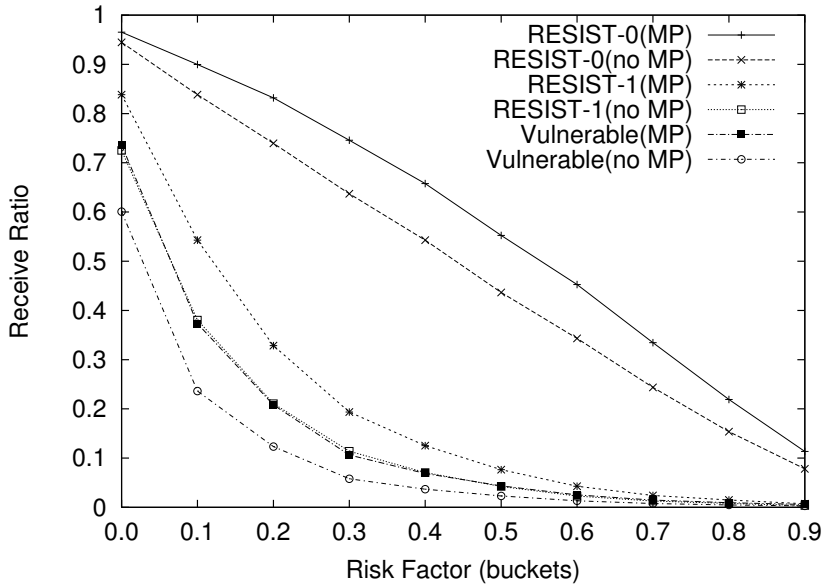


Figure 5: Using two paths (MP) instead of one increases the overall performance of the different strategies.

RESIST-0. In general, as the risk factor increases, the performance of routing protocols decreases. More importantly, for both vulnerable and RESIST-1 cases, this decrease is roughly exponential, whereas for RESIST-0, it has a better, linear decrease, as it does not allow nodes to lie about their distance to the sink.

Fig. 4 confirms that when malicious sensors are able to lie, they can attract more network traffic and thus, incur a much higher impact in the WSN. The linear decrease in performance of RESIST-0 seems to be the upper bound of the performance we can obtain by only addressing the sinkhole attacks. To get better results, one must also fight selective forwarding attacks. An attractive approach to decrease the impact of selective-forwarding attacks is to send each message through multiple paths to the sink. Fig. 5 shows the improvement gained by using two paths per message (one FTree path and another RRobin path), for different resiliency levels. An improvement of 5% is observed in all cases, except for RESIST-1 where it can reach 10%. Hence, in the vulnerable case, even if more than two paths is used, this still would not be sufficient to reach the performance of RESIST-0.

5.3 Routing Protocols Comparison

In this section, we compare the three routing protocols for resilient (i.e., RESIST-0 and RESIST-1) and vulnerable cases. To make such a comparison, we also modified the computation of the risk factor to represent the attacker capability more accurately. The main goal of this study is to understand which routing protocol is more advantageous among the three.

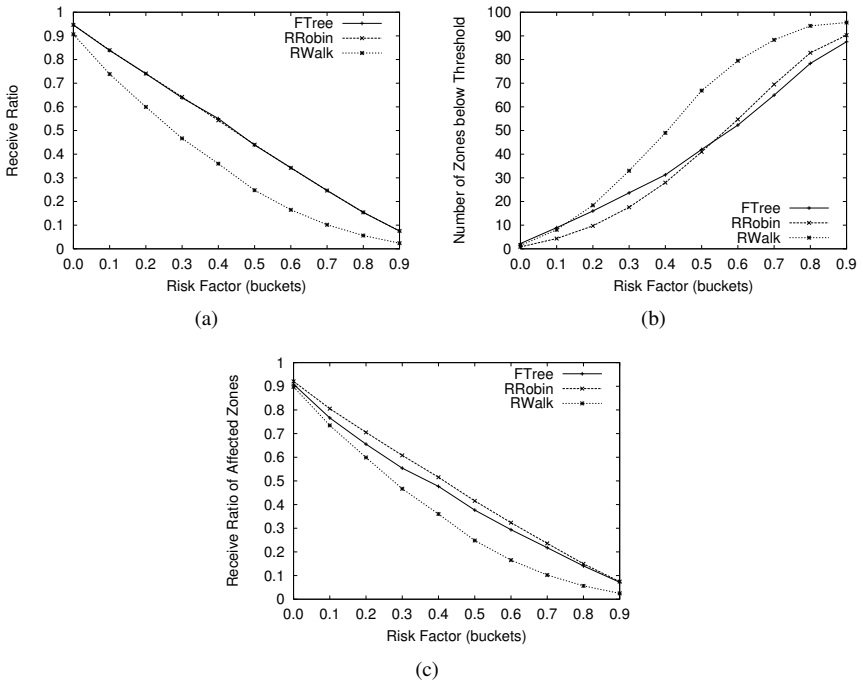


Figure 6: RESIST-0 performance evaluation: (a) Comparative performance of routing protocols. (b) Number of severely affected zones in the network (threshold =60%). (c) Received ratio of affected zones in the network.

5.3.1 Performance in RESIST-0 Case

The performance results with RESIST-0 is depicted in Fig. 6(a). Note that even if sinkhole attacks are avoided, malicious nodes can still perform the selective forwarding attack. Fig. 6(a) clearly shows that *FTree* and *RRobin* outperform *RWalk*. This is expected as in *RWalk*, the average path length that each message travels to the sink is longer. This consequently increases the probability of meeting a malicious node on the path. Further experimentation on *RWalk* also showed that the protocol performance is inversely proportional to n. This actually means that the best case for n-hop random walk is achieved when $n = 0$, in which case *RWalk* is equivalent to *FTree* routing.

Fig. 6(a) also shows that *FTree* and *RRobin* have similar performance. This is surprising since, intuitively, the performance of *RRobin* should be better compared to *FTree*. Analyzing the results, we observe that, as expected, for sensors, which have malicious parents, *RRobin* improves the performance by letting these nodes periodically send to alternative parents. However, this does not necessarily improve overall performance as *the reverse case* also holds: sensor nodes with good parents on the routing tree use malicious nodes as parents in a round robin fashion. Consequently, any gain from *RRobin* is neutralized by putting sensor nodes with good parents at risk.

To understand the effect of malicious nodes on the protocol behavior better, we divide the network into 100 equal zones and define the *failure threshold of a zone* as the percentage of data sent by the zone that needs to be dropped to qualify the zone as poorly monitored. In reality, this threshold would depend on the criticality of the sensor network application. We set the failure threshold as 60% in our experiments. Fig. 6(b)

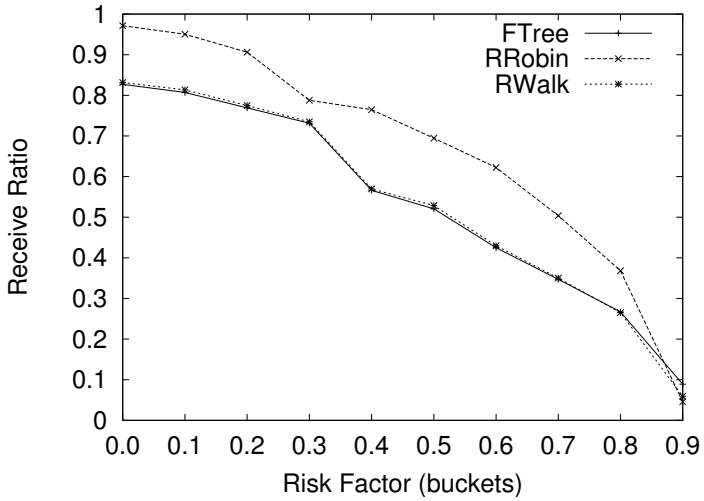


Figure 7: Comparative performance of routing protocols without any resistant reconfiguration protocol (using *Vulnerable Risk Factor*).

illustrates how many zones fell below the failure threshold for each risk factor bucket and routing protocol. Initially, the number of zones below failure threshold is higher for *FTree* than *RRobin*. Coupled with the fact that both protocols share the same receive ratio (see Fig. 6(c)), this means that *RRobin* just diffuses the effect of malicious nodes to more zones, so that fewer zones actually fail. However, as the risk factor increases, the number of zones below threshold increases beyond *FTree* due to *the reverse case* appearing more often. Essentially, increasing the failure threshold moves the shift point to the right. Nevertheless, although the number of affected zones is higher for *RRobin*, the average received data ratio per affected zone still remains higher than *FTree*. On the other hand, the number of failed zones in *RWalk* is always the highest due to its overall poor performance.

5.3.2 Performance in RESIST-1 and vulnerable cases

To better understand the performance of RESIST-1 and vulnerable cases, we slightly modified the computation of the risk factor presented in Section 3. The main reason for this modification is to represent the different malicious power of compromised nodes in RESIST-1 and vulnerable cases. Note that the only difference between the cases is the advertised *distance to the sink*, the only change in the computation is the way initial distances are calculated for each sensor. In the RESIST-1 case, since a malicious node can only lie by one hop, its distance is equal to that of its neighbor with the smallest distance to the sink. For the vulnerable case, malicious nodes pretend to be the sink, and so, the distance of each malicious node is 0. Hence, the shortest path algorithm needs to be run once for each sink, real and pretend. At the end, each node is assigned the shortest distance sink of these runs. Hereafter, each version of the risk factor is referred as *RESIST-1 Risk Factor* and *Vulnerable Risk Factor*, respectively.

Figs. 7 and 8 show the performance of *FTree*, *RRobin* and *RWalk* ($n = 1$) under the RESIST-1 and vulnerable cases. For each graph, we partitioned topologies based on their respective risk factors (i.e., *RESIST-1* and *Vulnerable Risk Factor*). Quite different than the RESIST-0 results (see Fig. 6(a)), *RRobin* performs the best for the vulnerable

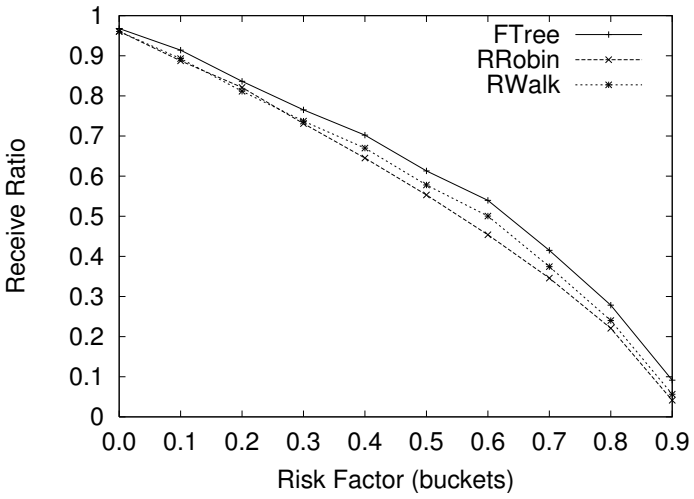


Figure 8: Comparative performance of routing protocols under the RESIST-1 reconfiguration protocol (using *RESIST-1 Risk Factor*).

case (see Fig. 7). This is because, in this case, the *FTree* algorithm outputs a forest of small routing trees, where the real sink and each malicious node is the root of one of these trees. Obviously, only the nodes that belong to the tree of the real sink can deliver data. In contrast, since *RRobin* allows nodes to follow different routes to the sink, it is able to reduce the effect of these sinkhole attacks. Note that *the reverse case* of *RRobin* (i.e., nodes with good parents use malicious nodes as alternative parents) does still exist. However, in the vulnerable case, the effect of fragmenting the network into several trees with *FTree* is greater than *the reverse case* of *RRobin*. Such fragmentation also occurs in *RWalk*, which explains why its performance is lower than *RRobin* as well. Essentially, in *RWalk*, the routing tree is built in the same way as in *FTree*.

Interestingly, *RRobin* cannot sustain the same performance in the RESIST-1 case. In the vulnerable case, the performance of *FTree* and *RWalk* is devastated by the fragmentation of the network into disconnected trees. Hence, *RRobin* is able to perform better. However, note that, in *RRobin*, if a node receives the first `Hello` message from a malicious node, then other neighbors may not be able to join the set of alternative parents if they advertise longer distances. Hence, the set of alternative parents becomes very small and most often consists of one malicious parent (or one of its descendants). This problem, although it appears in the vulnerable case too, is more obviously seen in RESIST-1 case, since *FTree* and *RWalk* can perform better in this case. Hence, in the RESIST-1 case, all protocols perform comparably, with *FTree* performing slightly better (see Fig. 8).

6 Discussion

In this section, we discuss the cost of using cryptographic primitives in sensors networks, and the impact of collusion on RESIST protocols.

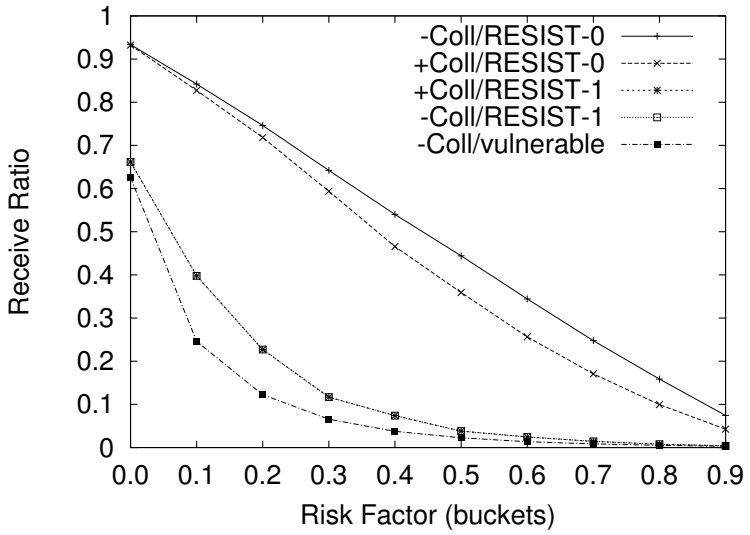


Figure 9: In the presence of short range collusion (+Coll), RESIST-0 still performs much better than vulnerable routing without collusion. RESIST-1 is not affected, since the default malicious behavior is similar to collusion.

6.1 Reducing the Cost of Cryptography for Sensors

The main cost of our RESIST protocols is the size of `Hello` messages, which carry multiple tokens that contain cryptographic values. Although, this might strain the significantly resource-constrained WSNs, we believe this can be avoided by an efficient way of implementing tokens. For instance, in [11] one-way hash functions are used so that a token T_{k+1} can be computed from a token T_k. Using this method, `Hello` messages need only to contain the first and the last token, signed by the sink, i.e. $\texttt{Hello}(H^k(T_0), (epoch, H^D(T_0))_{signed(K^{sink}_{pri})})$, where d is the diameter of the network. On receiving this message, a sensor can find its hop-count distance k by hashing $D-k$ times the token $H^k(T_0)$ until it reaches $H^D(T_0)$. Moreover, as it does not know T_0, and H is a one-way function, it cannot compute $H^{k-1}(T_0)$ and thus, it is not able to lie more than one hop (i.e., as in RESIST-1). Furthermore, recently, [15] showed that Elliptic Curves Cryptography (ECC) can be implemented at a very low cost in WSN. ECC keys are known to be much smaller than equivalent RSA keys [12], so that signatures and keys shorter than 110 bits would be largely sufficient in most contexts.

6.2 The Impact of Collusion on RESIST Protocols

To be able to implement sinkhole attacks in the presence of RESIST protocols, malicious sensors have to be designed to collaborate, to share good tokens (i.e., a token that can prove a short distance to the sink) they are able to collect during reconfigurations. Thus, RESIST protocols limit the power of collusion attacks to the closest distance an attacker can get to the sink. Hence, malicious nodes residing at the border of the network, for example, would not be able to disrupt it.

Collusion is also limited by the communication capabilities of malicious sensors: in Fig. 9, we simulated the impact of collusion when colluding sensors have normal radio ranges and are distributed randomly on the network area. In our simulations, malicious nodes exchange tokens so that they all appear the same distance to the sink (i.e., the distance of the malicious node that is closest to the sink). Our results show that the performance of RESIST-1 is not affected by the presence of colluding nodes. This was expected, as the collusion among malicious nodes do not necessarily create a higher impact on the structure of the tree. On the other hand, in RESIST-0, sharing of tokens enables replying challenges for shorter distances and hence, has an effect on performance.

Finally, the most dreadful attack would be a malicious sensor, close to the sink with a long radio range, allowing it to propagate a very good token to malicious sensors far from the sink. However, collusion in this case might not cause a significantly higher degradation in performance, as the dominant impact already comes from the malicious node closest to the sink.

7 Related work

Security in wireless networks is attracting the attention of many researchers [25] since security is vital to guarantee correct operation of sensor protocols. Nevertheless, this is also an extremely challenging task as it is relatively easy to launch an attack in a sensor network due to the wireless communication medium. The main conclusion of recent studies on secure routing is that updating current protocols with security extensions is not sufficient and that routing protocols should be designed from scratch with security in mind.

This paper focuses particularly on sink-hole and selective forwarding attacks. Most other approaches against these attacks revolves around detection of malicious nodes [24, 19, 11]. In [24], multi-hop acknowledgments are used to detect and blacklist nodes that do selective forwarding attacks. However, in addition to its cost, the proposed scheme requires geographical location information and strict synchronization. In [18, 19], a learning technique based on neural networks is used to predict the sensor measurements, and a reputation scheme is used to mark nodes as faulty if their reports are too different from predictions. In [11], a protocol similar to RESIST-1 is proposed, but without strong cryptography. As a consequence, it requires a protocol to detect malicious sensors (reports are vulnerable to falsification) and to blacklist nodes (through a complex messaging mechanism).

An interesting analysis of DDoS attacks in sensor networks, which also takes into account different network parameters and some counter measures, is presented in [1]. While their work covers TCP JellyFish and selective-forwarding attacks, we focus on sinkhole attacks. Moreover, our study of the Risk Factor metric captures more network characteristics.

An intuitive approach against selective forwarding attacks is to use multipath routing [8, 6]. However, such a protocol dramatically increases communication overhead as the redundancy of paths increases. In addition, these paths eventually converge to a few nodes surrounding the base station where malicious nodes can have a dreadful impact. Indeed, our simulation results show that the efficiency of this approach is limited, as confirmed by [1].

Trust-based systems [17, 16, 21] are interesting approaches to deal with selective forwarding attacks. In these systems, interactions between sensors are used for trust

level computation. Such systems are, however, often complex. We believe resilience, as provided by our protocols, is a better choice. As in [13], RESIST could use trust levels in the choice of the set of nodes to be considered during the round robin procedure.

8 Conclusion

In this paper, we introduced a new metric, the Risk Factor, to measure the impact of selective forwarding and sinkhole attacks on sensor networks. We showed that it successfully captures different topology-based parameters, such as the position and number of malicious nodes, the network scale, and attacker capability. Our performance evaluation confirmed that the resilience of our protocols is high even in the presence of collusions. We then presented two reconfiguration protocols that increase the resilience of the network in the presence of sink-hole attacks: RESIST-1 prevents malicious nodes from lying about their distance to the sink more than one hop; RESIST-0, which although being more expensive to use, completely stops malicious nodes from lying about their distance. Our performance evaluation confirmed the higher resilience of our protocols, even in the presence of some collusion. Nevertheless, as future work, we plan to evaluate the computation overhead of our schemes and to provide more collusion-resistance into our protocols.

References

[1] I. Aad, J.-P. Hubaux, and E. W. Knightly. Impact of denial of service attacks on ad hoc networks. *IEEE/ACM Trans. on Networking*, 2008.

[2] Z. Bar-Yossef, R. Friedman, and G. Kliot. RaWMS - random walk based lightweight membership service for wireless ad hoc networks. *ACM ACM Transactions on Computer Systems (TOCS)*, 26(2):1–66, June 2008.

[3] Ugur Cetintemel, Andrew Flinders, and Ye Sun. Power-efficient data dissemination in wireless sensor networks. In *ACM MobiDE*, 2003.

[4] P. Gupta and P. Kumar. Critical power for asymptotic connectivity in wireless networks. In *Stochastic Analysis, Control, Optimization and Applications*, 1998.

[5] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *ACM MOBICOM*, 2000.

[6] Shivakant Mishra Jing Deng, Richard Han. Insens: Intrusion-tolerant routing in wireless sensor networks. In *IEEE ICDCS*, 2003.

[7] W. R. Pires Junior, T. H. de P. Figueiredo, and Hao Chi Wong. Malicious node detection in wireless sensor networks. In *IEEE IPDPS*, 2004.

[8] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *1st IEEE Workshop on Sensor Network Protocols and Applications (SNPA)*, 2003.

[9] Bhaskar Krishnamachari, Deborah Estrin, and Stephen Wicker. The impact of data aggregation in wireless sensor networks. In *IEEE ICDCS*, 2002.

[10] Fabrice Le Fessant, Cigdem Sengul, and Anne-Marie Kermarrec. Pace-maker: Tracking peer availability in large networks. Technical Report RR-6594, INRIA, 2008.

[11] Suk-Bok Lee and Yoon-Hwa Choi. A secure alternate path routing in sensor networks. *Computer Communications, Elsevier*, 30, 2006.

[12] Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. In *PKC*, 2000.

[13] Zhaoyu Liu, A.W. Joy, and R.A. Thompson. A dynamic trust model for mobile ad hoc networks. In *Future Trends of Dist. Computing Systems*, 2004.

[14] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks. *ACM OSR*, 2002.

[15] David J. Malan, Matt Welsh, and Michael D. Smith. Implementing public-key infrastructure for sensor networks. *ACM Trans. Sen. Netw.*, 4(4):1–23, 2008.

[16] S. Marti, T.J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *ACM MOBICOM*, 2000.

[17] D. H. McKinight and A. E. Kamal. The meanings of trust. Technical report, Univ. Minnesota, 1996.

[18] Partha Mukherjee and Sandip Sen. Detecting malicious sensor nodes from learned data patterns. In *Agent Technology for Sensor Networks*, 2007.

[19] Partha Mukherjee and Sandip Sen. Using learned data patterns to detect malicious nodes in sensor networks. In *ICDCN*, 2008.

[20] Leonardo B. Oliveira, Diego Aranha, Eduardo Morais, Felipe Daguano, Julio López, and Ricardo Dahab. Identity-based encryption for sensor networks. In *IEEE PERCOM*, 2007.

[21] Y. Sun, Z. Han, W. Yu, and K. J. R. Liu. A trust evaluation framework in distributed networks: Vulnerability analysis and defense against attacks. In *IEEE INFOCOM*, 2007.

[22] Fan Ye, Alvin Chen, Songwu Lu, and Lixia Zhang. Gradient broadcast: A robust, long-live large sensor network. Technical report, UCLA, 2001.

[23] Fan Ye, Alvin Chen, Songwu Lu, and Lixia Zhang. A scalable solution to minimum cost forwarding in large sensor networks. In *Conf. on Computer Communications and Networks*, 2001.

[24] Bo Yu and Bin Xao. Detecting selective forwarding attacks in wireless sensor networks. In *IEEE IPDPS*, 2006.

[25] W. Yu and K.J.R. Liu. Attack-resistant cooperation stimulation in autonomous ad hoc networks. *IEEE/ACM Trans. on Networking*, 2005.

[26] Yonggang Jerry Zhao, Ramesh Govindan, and Deborah Estrin. Residual energy scans for monitoring wireless sensor networks. *IEEE WCNC*, 2002.

[27] Seyit A. Çamtepe and Bulent Yener. Key distribution mechanisms for wireless sensor networks: a survey. *IEEE/ACM Trans. on Networking*, 2005.

Contents

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 Orsay Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399